High Contrast Mode

Play+ High Contrast Mode: Accessibility by Default Introduction ■ For many users, accessibility isn't a preference—it's a necessity. High Contrast Mode is a system-level feature (like Windows High Contrast or macOS's Increase contrast) that enhances visual clarity for users with low vision by increasing the contrast between foreground and background elements. In the Play+ ecosystem, supporting High Contrast Mode isn't an optional enhancement—it's a default promise . As part of our Inclusive design pillar, Play+ ensures that applications are not only accessible, but also effortless to navigate for users with visual impairments. No configuration needed. No special themes to maintain. Just built-in, adaptive accessibility. How It Works: Zero Effort, Maximum Clarity ■ High Contrast Mode in Play+ is designed to just work : User Enables High Contrast Mode Users activate it in their OS settings (e.g., Windows or macOS). Play+ Detects the Setting The theming engine uses CSS media queries such as: @media (prefers-contrast : more) { ... ; } or legacy support like -ms-high-contrast: active for Internet Explorer/Edge. A High Contrast Theme Is Applied Automatically A predefined set of high-contrast styles takes over, overriding both the default and dark themes. You don't need to create or manage a separate High Contrast theme—Play+ handles it entirely. The Derivation Logic: Designed for Legibility ■ To maximize legibility, the Play+ engine applies these key transformations: Simplified Colors Decorative and brand colors are removed. Backgrounds become solid black; text becomes white, yellow, or green—matching the user's OS-level palette. Borders Become Structural Elements that rely on color contrast now use clear, solid borders for separation: Buttons Form Inputs Cards Navigation Elements Shadows and Gradients Removed All shadows, gradients, and other decorative styles are stripped out to remove visual noise. Focus Indicators Are Amplified The default focus ring is replaced with a prominent solid outline for easier keyboard navigation. Implementation Details ■ CSS Media Query Detection ■ The high contrast mode is automatically detected using CSS media queries: /* Modern browsers */ @media (prefers-contrast : more) { /* High contrast styles */ } /* Legacy IE/Edge support */ @media (-ms-high-contrast : active) { /* High contrast styles */ } Theme File Structure ■ The high contrast theme is implemented in: projects/play-comp-library/src/lib/styles/tokens/themes/_high-contrast.css Key Semantic Token Overrides ■ /* Background colors */ --color-background-primary : #000000 ; --color-background-secondary : #1a1a1a ; /* Text colors */ --color-text-primary : #ffffff ; --color-text-secondary : #ffff00 ; --color-text-interactive : #00ff00 ; /* Border colors */

--color-border-default : #ffffff ; --color-border-interactive : #ffff00 ; --color-border-focus : #00ff00 ; /* Focus indicators */ --focus-ring-color : #00ff00 ; --focus-ring-width : 3 px ; Component Adaptation ■ Buttons ■ Solid borders replace background colors Text becomes high contrast white/yellow Focus indicators are prominently displayed Form Elements ■ Input borders become solid white Labels maintain high contrast Focus states are clearly visible Cards and Containers ■ Backgrounds become solid black Borders provide structural separation Content remains highly legible Navigation ■ Links use high contrast colors Active states are clearly indicated Focus navigation is enhanced Testing High Contrast Mode ■ Manual Testing ■ Enable High Contrast Mode in your OS settings Navigate to the High Contrast Test page: /high-contrast-test Verify all components adapt appropriately Test keyboard navigation and focus indicators Automated Testing ■ Use the provided CLI script for quick testing: npm run test:high-contrast Browser Developer Tools ■ Open Chrome DevTools Go to Rendering tab Enable "Emulate a focused page" Check "Forced colors" to simulate high contrast Respecting User Intent: No Overrides Allowed ■ Unlike our light/dark themes, High Contrast Mode in Play+ is non-configurable . Why? Because when users enable this setting at the OS level, they are expressing a deliberate accessibility need. Overriding it with brand colors or design embellishments would compromise usability—and violate the spirit of inclusive design. Play+ respects user agency by fully honoring system preferences without interference. Developer Checklist ■ Even though High Contrast Mode is automatic, we recommend manual testing for critical workflows. Enable High Contrast Mode on your OS (Windows) or "Increase contrast" (macOS). Navigate to /high-contrast-test to see all components in high contrast mode. Verify that all interactive elements (buttons, inputs, links) are clearly distinguishable. Ensure that color is never the only indicator for statuses or alerts—add icons or text labels. Check that the focus indicator is clearly visible when tabbing through the UI. Confirm that all text and icons have sufficient contrast and legibility. Test with screen readers to ensure proper ARIA support. Validate that form validation messages are clearly visible. Integration with Angular Components ■ Automatic Detection ■ The high contrast mode is automatically detected in Angular components: ngOnInit ( ) { // Check for high contrast mode this . checkHighContrastMode ( ) ; } checkHighContrastMode ( ) { const mediaQuery = window . matchMedia ('(prefers-contrast: more)') ; this . isHighContrastActive = mediaQuery . matches ; // Listen for changes mediaQuery . addEventListener ('change' , (e) => { this . isHighContrastActive = e . matches ; }) ; } Component Usage ■ All Play+ components automatically adapt to high contrast mode: <!-- These components automatically adapt --> < ava-button variant = " primary " label = " Primary Button " > </ ava-button > < ava-textbox label = " Input Field " placeholder = " Enter text " > </ ava-textbox

> < ava-card > < div class = " card-content " > Content automatically adapts </ div > </ ava-card > Browser Support ■ Browser Version Support Chrome 76+ Full support Firefox 70+ Full support Safari 13+ Full support Edge 79+ Full support IE 11 Legacy support Troubleshooting ■ High Contrast Mode Not Detected ■ Ensure the theme CSS file is properly imported Check that the media query is correctly formatted Verify OS settings are properly configured Components Not Adapting ■ Check that components are using semantic CSS tokens Ensure no hardcoded colors are overriding the theme Verify the component is properly importing the theme styles Focus Indicators Not Visible ■ Check that focus styles are not being overridden Ensure proper CSS specificity for focus states Verify that interactive elements are properly focusable Summary ■ High Contrast Mode in Play+ is more than a feature—it's a commitment. By stripping UI down to its clearest, most essential form, Play+ ensures that users with visual challenges can navigate and understand your application with confidence and ease. And because it's automatic , developers don't have to do anything extra to support it—just build, and Play+ takes care of the rest. Inclusive by default. Accessible by design.

```
@media (prefers-contrast: more) {
  ...;
}


---

/* Modern browsers */
@media (prefers-contrast: more) {
  /* High contrast styles */
}

/* Legacy IE/Edge support */
@media (-ms-high-contrast: active) {
  /* High contrast styles */
}


---

projects/play-comp-library/src/lib/styles/tokens/themes/_high-contrast.css


---

/* Background colors */
--color-background-primary: #000000;
--color-background-secondary: #1a1a1a;

/* Text colors */
```

```
--color-text-primary: #ffffff;
--color-text-secondary: #ffff00;
--color-text-interactive: #00ff00;

/* Border colors */
--color-border-default: #ffffff;
--color-border-interactive: #ffff00;
--color-border-focus: #00ff00;

/* Focus indicators */
--focus-ring-color: #00ff00;
--focus-ring-width: 3px;
```

```
npm run test:high-contrast
```

```
ngOnInit() {
  // Check for high contrast mode
  this.checkHighContrastMode();
}

checkHighContrastMode() {
  const mediaQuery = window.matchMedia('(prefers-contrast: more)');
  this.isHighContrastActive = mediaQuery.matches;

  // Listen for changes
  mediaQuery.addEventListener('change', (e) => {
    this.isHighContrastActive = e.matches;
  });
}
```

```
<!-- These components automatically adapt -->
<ava-button variant="primary" label="Primary Button"></ava-button>
<ava-textbox label="Input Field" placeholder="Enter text"></ava-textbox>
<ava-card>
  <div class="card-content">Content automatically adapts</div>
</ava-card>
```